A New Algorithm for the Alignment of Phonetic Sequences

Grzegorz Kondrak
Department of Computer Science
University of Toronto
Toronto, Ontario, Canada M5S 3G4
kondrak@cs.toronto.edu

Abstract

Alignment of phonetic sequences is a necessary step in many applications in computational phonology. After discussing various approaches to phonetic alignment, I present a new algorithm that combines a number of techniques developed for sequence comparison with a scoring scheme for computing phonetic similarity on the basis of multivalued features. The algorithm performs better on cognate alignment, in terms of accuracy and efficiency, than other algorithms reported in the literature.

1 Introduction

Identification of the corresponding segments in sequences of phones is a necessary step in many applications in both diachronic and synchronic phonology. Usually we are interested in aligning sequences that represent forms that are related in some way: a pair of cognates, or the underlying and the surface forms of a word, or the intended and the actual pronunciations of a word. Alignment of phonetic sequences presupposes transcription of sounds into discrete phonetic segments, and so differs from matching of utterances in speech recognition. On the other hand, it has much in common with the alignment of proteins and DNA sequences. Many methods developed for molecular biology can be adapted to perform accurate phonetic alignment.

Alignment algorithms usually contain two main components: a metric for measuring distance between phones, and a procedure for finding the optimal alignment. The former is often calculated on the basis of phonological features that encode certain properties of phones. An obvious candidate for the latter is a well-known dynamic programming (DP) algorithm for string alignment (Wagner and Fischer, 1974), although other algorithms can used as well. The task of finding the optimal alignment is closely linked to the task of calculating the distance between two sequences. The basic DP algorithm accomplishes both tasks. Depending on the application, either of the results, or both, can be used.

Within the last few years, several different approaches to phonetic alignment have been reported. Covington (1996) used depth-first search and a special distance function to align words for historical comparison. In a follow-up paper (Covington, 1998), he extended the algorithm to align words from more than two languages. Somers (1998) proposed a special algorithm for aligning children's articulation data with the adult model. Gildea and Jurafsky (1996) applied the DP algorithm to pre-align input and output phonetic strings in order to improve the performance of their transducer induction system. Nerbonne and Heeringa (1997) employed a similar procedure to compute relative distance between words from various Dutch dialects. Some characteristics of these implementations are juxtaposed in Table 1.

In this paper, I present a new algorithm for the alignment of cognates. It combines various techniques developed for sequence comparison with an appropriate scoring scheme for computing phonetic similarity on the basis of multivalued features. The new algorithm performs better, in terms of accuracy and efficiency, than comparable algorithms reported by Covington (1996) and Somers (1999). Although the main focus of this paper is diachronic phonology, the techniques proposed here can also be applied in other contexts where it is necessary to align phonetic sequences.

2 Comparing Phones

To align phonetic sequences, we first need a function for calculating the distance between individual phones. The numerical value assigned by the function to a pair of segments is referred to as the cost, or penalty, of substitution. The function is often extended to cover pairs consisting of a segment and the null character, which correspond to the opera-

Algorithm	Calculation of alignment	Calculation of distance	Dynamic programming	Phonological features
Covington (1996)	explicit	implicit	no	no
Somers (1998)	explicit	no	no	multivalued
Nerbonne and Heeringa (1997)	implicit	explicit	yes	binary
Gildea and Jurafsky (1996)	explicit	implicit	yes	binary

Table 1: Comparison of alignment algorithms.

tions of insertion and deletion (also called *indels*). A distance function that satisfies the following axioms is called a *metric*:

1. $\forall a,b : d(a,b) \geq 0$ (*nonnegative property*)
2. $\forall a,b : d(a,b) = 0 \Leftrightarrow a = b$ (*zero property*)
3. $\forall a,b : d(a,b) = d(b,a)$ (*symmetry*)
4. $\forall a,b,c : d(a,b) + d(b,c) \geq d(a,c)$ (*triangle inequality*)

2.1 Covington's Distance Function vs. Feature-Based Metrics

Covington (1996), for his cognate alignment algorithm, constructed a special distance function. It was developed by trial and error on a test set of 82 cognate pairs from various related languages. The distance function is very simple; it uses no phonological features and distinguishes only three types of segments: consonants, vowels, and glides. Many important characteristics of sounds, such as place or manner of articulation, are ignored. For example, both *yacht* and *will* are treated identically as a glide-vowel-consonant sequence. The function's values for substitutions are listed in the "penalty" column in Table 2. The penalty for an indel is 40 if it is preceded by another indel, and 50 otherwise. Covington (1998) acknowledges that his distance function is "just a stand-in for a more sophisticated, perhaps feature-based, system".[1]

Both Gildea and Jurafsky (1996) and Nerbonne and Heeringa (1997) use distance functions based on binary features. Such functions have the ability to distinguish a large number of different phones. The underlying assumption is that the number of binary features by which two given sounds differ is a good indication of their proximity. Phonetic segments are represented by binary vectors in which every entry stands for a single articulatory feature. The penalty for a substitution is defined as the Hamming distance between two feature vectors. The penalty for indels is established more or less arbitrarily.[2] A distance function defined in such a way satisfies all metric axioms.

It is interesting to compare the values of Covington's distance function with the average Hamming distances produced by a feature-based metric. Since neither Gildea and Jurafsky (1996) nor Nerbonne and Heeringa (1997) present their feature vectors in sufficient detail to perform the calculations, I adopted a fairly standard set of 17 binary features from Hartman (1981).[3] The average feature distances between pairs of segments corresponding to every clause in Covington's distance function are given in Table 2, next to Covington's "penalties". By definition, the Hamming distance between identical segments is zero. The distance between the segments covered by clause #3 is also constant and equal to one (the feature in question being [long] or [syllabic]). The remaining average feature distances were calculated using a set of most frequent phonemes represented by 25 letters of the Latin alphabet (all but *q*). In order to facilitate comparison, the rightmost column of Table 2 contains the average distances interpolated between the minimum and the maximum value of Covington's distance function. The very high correlation (0.998) between Covington's penalties and the average distances demonstrates that feature-based phonology provides a theoretical basis for Covington's manually constructed distance function.

[1] Covington's distance function is not a metric. The zero property is not satisfied because the function's value for two identical vowels is greater than zero. Also, the triangle inequality does not hold in all cases; for example: $p(e,i) = 30$ and $p(i,y) = 10$, but $p(e,y) = 100$, where $p(x,y)$ is the penalty for aligning [x] with [y].

[2] Nerbonne and Heeringa (1997) fix the penalty for indels as half the average of the values of all substitutions. Gildea and Jurafsky (1996) set it at one fourth of the maximum substitution cost.

[3] In order to handle all the phones in Covington's data set, two features were added: [tense] and [spread glottis].

	Clause in Covington's distance function	Covington's penalty	Average Hamming distance	Interpolated average distance
1	*"identical consonants or glides"*	0	0.0	0.0
2	*"identical vowels"*	5	0.0	0.0
3	*"vowel length difference only"*	10	1.0	12.4
4	*"non-identical vowels"*	30	2.2	27.3
5	*"non-identical consonants"*	60	4.81	58.1
6	*"no similarity"*	100	8.29	100.0

Table 2: The clause-by-clause comparison of Covington's distance function (column 3) and a feature-based distance function (columns 4 and 5).

2.2 Binary vs. Multivalued Features

Although binary features are elegant and widely used, they might not be optimal for phonetic alignment. Their primary motivation is to classify phonological oppositions rather than to reflect the phonetic characteristics of sounds. In a strictly binary system, sounds that are similar often differ in a disproportionately large number of features. It can be argued that allowing features to have several possible values results in a more natural and phonetically adequate system. For example, there are many possible places of articulation, which form a near-continuum ranging from [labial] to [glottal].

Ladefoged (1995) devised a phonetically-based multivalued feature system. This system has been adapted by Connolly (1997) and implemented by Somers (1998). It contains about 20 features with values between 0 and 1. Some of them can take as many as ten different values (e.g. [place]), while others are basically binary oppositions (e.g. [nasal]). Table 3 contains examples of multivalued features.

The main problem with both Somers's and Connolly's approaches is that they do not differentiate the weights, or *saliences*, that express the relative importance of individual features. For example, they assign the same salience to the feature [place] as to the feature [aspiration], which results in a smaller distance between [p] and [k] than between [p] and [p^h]. I found that in order to avoid such incongruous outcomes, the salience values need to be carefully differentiated; specifically, the features [place] and [manner] should be assigned significantly higher saliences than other features (the actual values used in my algorithm are given in Table 4). Nerbonne and Heeringa (1997) experimented with weighting each feature by information gain but found it had an adverse effect on the quality of the alignments. The question of how to derive salience values in a principled manner is still open.

2.3 Similarity vs. Distance

Although all four algorithms listed in Table 1 measure relatedness between phones by means of a *distance* function, such an approach does not seem to be the best for dealing with phonetic units. The fact that Covington's distance function is not a metric is not an accidental oversight; rather, it reflects certain inherent characteristics of phones. Since vowels are in general more volatile than consonants, the preference for matching identical consonants over identical vowels is justified. This insight cannot be expressed by a metric, which, by definition, assigns a zero distance to all identical pairs of segments. Nor is it certain that the triangle inequality should hold for phonetic segments. A phone that has two different places of articulation, such as labio-velar [w], can be close to two phones that are distant from each other, such as labial [b] and velar [g].

In my algorithm, below, I employ an alternative approach to comparing segments, which is based on the notion of *similarity*. A similarity scoring scheme assigns large positive scores to pairs of related segments; large negative scores to pairs of dissimilar segments; and small negative scores to indels. The optimal alignment is the one that maximizes the overall score. Under the similarity approach, the score obtained by two identical segments does not have to be constant. Another important advantage of the similarity approach is the possibility of performing *local alignment* of phonetic sequences, which is discussed in the following section.

3 Tree Search vs. Dynamic Programming

Once an appropriate function for measuring similarity between pairs of segments has been designed,

Feature name	Phonological term	Numerical value
Place	[bilabial]	1.0
	[labiodental]	0.95
	[dental]	0.9
	[alveolar]	0.85
	[retroflex]	0.8
	[palato-alveolar]	0.75
	[palatal]	0.7
	[velar]	0.6
	[uvular]	0.5
	[pharyngeal]	0.3
	[glottal]	0.1
Manner	[stop]	1.0
	[affricate]	0.9
	[fricative]	0.8
	[approximant]	0.6
	[high vowel]	0.4
	[mid vowel]	0.2
	[low vowel]	0.0
High	[high]	1.0
	[mid]	0.5
	[low]	0.0
Back	[front]	1.0
	[central]	0.5
	[back]	0.0

Table 3: Multivalued features and their values.

we need an algorithm for finding the optimal alignment of phonetic sequences. While the DP algorithm, which operates in quadratic time, seems to be optimal for the task, both Somers and Covington opt for exhaustive search strategies. In my opinion, this is unwarranted.

Somers's algorithm is unusual because the selected alignment is not necessarily the one that minimizes the sum of distances between individual segments. Instead, it recursively selects the most similar segments, or "anchor points", in the sequences being compared. Such an approach has a serious flaw. Suppose that the sequences to be aligned are *tewos* and *divut*. Even though the corresponding segments are slightly different, the alignment is straightforward. However, an algorithm that looks for the best matching segments first, will erroneously align the two *t*'s. Because of its recursive nature, the algorithm has no chance of recovering from such an error.[4]

[4]The criticism applies regardless of the method of choosing the best matching segments (see also Section 5).

Syllabic	5	Place	40
Voice	10	Nasal	10
Lateral	10	Aspirated	5
High	5	Back	5
Manner	50	Retroflex	10
Long	1	Round	5

Table 4: Features used in ALINE and their salience settings.

Covington, who uses a straightforward depth-first search to find the optimal alignment, provides the following arguments for eschewing the DP algorithm.

> First, the strings being aligned are relatively short, so the efficiency of dynamic programming on long strings is not needed. Second, dynamic programming normally gives only one alignment for each pair of strings, but comparative reconstruction may need the *n* best alternatives, or all that meet some criterion. Third, the tree search algorithm lends itself to modification for special handling of metathesis or assimilation.[5] (Covington, 1996)

The efficiency of the algorithm might not be relevant in the simple case of comparing two words, but if the algorithm is to be of practical use, it will have to operate on large bilingual wordlists. Moreover, combining the alignment algorithm with some sort of strategy for identifying cognates on the basis of phonetic similarity is likely to require comparing thousands of words against one another. Having a polynomially bound algorithm in the core of such a system is crucial. In any case, since the DP algorithm involves neither significantly larger overhead nor greater programming effort, there is no reason to avoid using it even for relatively small data sets.

The DP algorithm is also sufficiently flexible to accommodate most of the required extensions without compromising its polynomial complexity. A simple modification will produce all alignments that are within ε of the optimal distance (Myers, 1995). By applying methods from the operations research literature (Fox, 1973), the algorithm can be adapted to deliver the *n* best solutions. Moreover, the basic set of editing operations (substitutions and indels)

[5]Covington does not elaborate on the nature of the modifications.

can be extended to include both transpositions of adjacent segments (metathesis) (Lowrance and Wagner, 1975) and compressions and expansions (Oommen, 1995). Other extensions of the DP algorithm that are applicable to the problem of phonetic alignment include affine gap scores and local comparison.

The motivation for generalized gap scores arises from the fact that in diachronic phonology not only individual segments but also entire morphemes and syllables are sometimes deleted. In order to take this fact into account, the penalty for a gap can be calculated as a function of its length, rather than as a simple sum of individual deletions. One solution is to use an *affine* function of the form $gap(x) = r + sx$, where r is the penalty for the introduction of a gap, and s is the penalty for each symbol in the gap. Gotoh (1982) describes a method for incorporating affine gap scores into the DP alignment algorithm. Incidentally, Covington's penalties for indels can be expressed by an affine gap function with $r = 10$ and $s = 40$.

Local comparison (Smith and Waterman, 1981) is made possible by using both positive and negative similarity scores. In local, as opposed to global, comparison, only similar subsequences are matched, rather than entire sequences. This often has the beneficial effect of separating inflectional and derivational affixes from the roots. Such affixes tend to make finding the proper alignment more difficult. It would be unreasonable to expect affixes to be stripped before applying the algorithm to the data, because one of the very reasons to use an automatic aligner is to avoid analyzing every word individually.

4 The algorithm

Many of the ideas discussed in previous sections have been incorporated into the new algorithm for the alignment of phonetic sequences (ALINE). Similarity rather than distance is used to determine a set of best local alignments that fall within ε of the optimal alignment.[6] The set of operations contains insertions/deletions, substitutions, and expansions/compressions. Multivalued features are employed to calculate similarity of phonetic segments. Affine gaps were found to make little difference when local comparison is used and they were subsequently removed from ALINE.[7] The algorithm has been implemented in C++ and will be made available in the near future.

```
algorithm Alignment
    input: phonetic sequences x and y
    output: alignment of x and y
    define S(i, j) = −∞ when i < 0 or j < 0

    for i ← 0 to |x| do
        S(i, 0) ← 0
    for j ← 0 to |y| do
        S(0, j) ← 0
    for i ← 1 to |x| do
        for j ← 1 to |y| do
            S(i, j) ← max(
                S(i−1, j) + σ_skip(x_i),
                S(i, j−1) + σ_skip(y_j),
                S(i−1, j−1) + σ_sub(x_i, y_j),
                S(i−1, j−2) + σ_exp(x_i, y_{j−1} y_j),
                S(i−2, j−1) + σ_exp(x_{i−1} x_i, y_j),
                0)

    T ← (1 − ε) × max_{i,j} S(i,j)

    for i ← 1 to |x| do
        for j ← 1 to |y| do
            if S(i, j) > T then
                Retrieve(i, j, 0)
```

Figure 1: The algorithm for computing the alignment of two phonetic sequences.

Figure 1 contains the main components of the algorithm. First, the DP approach is applied to compute the similarity matrix S using the σ scoring functions. The optimal score is the maximum entry in the whole matrix. A recursive procedure *Retrieve* (Figure 2) is called on every matrix entry that exceeds the threshold score T. The alignments are retrieved by traversing the matrix until a zero entry is encountered. The scoring functions for indels, substitutions and expansions are defined in Figure 3. C_{skip}, C_{sub}, and C_{exp} are the maximum scores for indels, substitutions, and expansions, respectively. C_{vwl} determines the relative weight of consonants and vowels. The default values are C_{skip} = –10, C_{sub} = 35, C_{exp} = 45 and C_{vwl} = 10. The *diff* function returns the difference between segments p and q for a given feature f. Set R_V contains features relevant for comparing two vowels: Syllabic, Nasal, Retroflex, High, Back, Round, and Long. Set

[6] Global and semiglobal comparison can also be used. In a semiglobal comparison, the leading and trailing indels are assigned a score of zero.

[7] They may be necessary, however, when dealing with languages that are rich in infixes.

procedure Retrieve(i, j, s)

if S(i, j) = 0 **then**
print(Out)
print("alignment score is *s*")
else
if S(i−1, j−1) + $\sigma_{sub}(x_i, y_j)$ + s $\geq$ T **then**
push(Out, "align x_i with y_j")
Retrieve(i−1, j−1, s + $\sigma_{sub}(x_i, y_j)$)
pop(Out)
if S(i, j−1) + $\sigma_{skip}(y_j)$ + s $\geq$ T **then**
push(Out, "align null with y_j")
Retrieve(i, j−1, s + $\sigma_{skip}(y_j)$)
pop(Out)
if S(i−1, j−2) + $\sigma_{exp}(x_i, y_{j-1}y_j)$ + s $\geq$ T **then**
push(Out, "align x_i with $y_{j-1}y_j$")
Retrieve(i−1, j−2, s + $\sigma_{exp}(x_i, y_{j-1}y_j)$)
pop(Out)
if S(i−1, j) + $\sigma_{skip}(x_j)$ + s $\geq$ T **then**
push(Out, "align x_i with null")
Retrieve(i−1, j, s + $\sigma_{skip}(x_j)$)
pop(Out)
if S(i−2, j−1) + $\sigma_{exp}(y_j, x_{i-1}x_i)$ + s $\geq$ T **then**
push(Out, "align $x_i x_{i-1}$ with y_j")
Retrieve(i−2, j−1, s + $\sigma_{exp}(y_j, x_{i-1}x_i)$)
pop(Out)

Figure 2: The procedure for retrieving alignments from the similarity matrix.

R_C contains features for comparing other segments: Syllabic, Manner, Voice, Nasal, Retroflex, Lateral, Aspirated, and Place. When dealing with double-articulation consonantal segments, only the nearest places of articulation are used. For a more detailed description of the algorithm see (Kondrak, 1999).

ALINE represents phonetic segments as vectors of feature values. Table 4 shows the features that are currently used by ALINE. Feature values are encoded as floating-point numbers in the range $[0.0, 1.0]$. The numerical values of four principal features are listed in Table 3. The numbers are based on the measurements performed by Ladefoged (1995). The remaining features have exactly two possible values, 0.0 and 1.0. A special feature 'Double', which has the same values as 'Place', indicates the second place of articulation. Thanks to its continuous nature, the system of features and their values can easily be adjusted and augmented.

5 Evaluation

The best alignments are obtained when local comparison is used. For example, when aligning English *grass* with Latin *grāmen*, it is important to match only the first three segments in each word; the remaining segments are unrelated. ALINE obviously does not know the particular etymologies, but it can make a guess because [s] and [m] are not very similar phonetically. Only local alignment is able to distinguish between the essential and non-essential correspondences in this case (Table 5).

$$\sigma_{skip}(p) = C_{skip}$$

$$\sigma_{sub}(p,q) = C_{sub} - \delta(p,q) - V(p) - V(q)$$

$$\sigma_{exp}(p,q_1q_2) = C_{exp} - \delta(p,q_1) - \delta(p,q_2) - V(p) - max(V(q_1),V(q_2))$$

where

$$V(p) = \begin{cases} 0 & \text{if } p \text{ is a consonant} \\ C_{vwl} & \text{otherwise} \end{cases}$$

$$\delta(p,q) = \sum_{f \in R} \text{diff}(p,q,f) \times \text{salience}(f)$$

where

$$R = \begin{cases} R_C & \text{if } p \text{ or } q \text{ is a consonant} \\ R_V & \text{otherwise} \end{cases}$$

Figure 3: Scoring functions.

The operations of compression and expansion prove to be very useful in the case of complex correspondences. For example, in the alignment of Latin *factum* with Spanish *hecho*, the affricate [ʧ] should be linked with both [k] and [t] rather than with just one of them, because it originates from the merger of the two consonants. Note that taking a se-

g	r	-	-	æ	s	
g	r	ā	m	e	n	
‖	g	r	æ	s	‖	
‖	g	r	ā	m	‖	en
‖	g	r	æ	‖	s	
‖	g	r	ā	‖	men	

Table 5: Three alignments of English *grass* and Latin *grāmen* obtained with global, semiglobal, and local comparison. The double bars delimit the aligned subsequences.

	Covington's alignments							*ALINE's alignments*					
three : trēs	θ	r	i	y				‖	θ	r	iy	‖	
	t	r	ē	s				‖	t	r	ē	‖	s
blow : flāre	b	l	-	-	o	w		‖	b	l	o	‖	w
	f	l	ā	r	e	-		‖	f	l	ā	‖	re
full : plēnus	f	-	-	-	u	l		‖	f	u	l	‖	
	p	l	ē	n	u	s		‖	p	-	l	‖	ēnus
fish : piscis	f	-	-	-	i	š		‖	f	i	š	‖	
	p	i	s	k	i	s		‖	p	i	s	‖	kis
I : ego	-	-	a	y				‖	ay	‖			
	e	g	o	-				‖	e	‖	go		
tooth : dentis	-	-	-	t	u	w	θ		‖	t	uw	θ	‖
	d	e	n	t	i	-	s	den	‖	t	i	s	‖

Table 6: Examples of alignments of English and Latin cognates.

quence of substitution and deletion as compression is unsatisfactory because it cannot be distinguished from an actual sequence of substitution and deletion. ALINE posits this operation particularly frequently in cases of diphthongization of vowels (see the alignments in Table 6).

Covington's data set of 82 cognates provides a convenient test for the algorithm. His English/Latin set is particularly interesting, because these two languages are not closely related. Some of the alignments produced by Covington's algorithm and ALINE are shown in Table 6. ALINE accurately discards inflectional affixes in *piscis* and *flāre*. In *fish/piscis*, Covington's aligner produces four alternative alignments, while ALINE selects the correct one. Both algorithms are technically wrong on *tooth/dentis*, but this is hardly an error considering that only the information contained in the phonetic string is available to the aligners. On Covington's Spanish/French data, ALINE does not make any mistakes. Unlike Covington's aligner, it properly aligns [l] in *arbol* with the second [r] in *arbre*. On his English/German data, it selects the correct alignment in those cases where Covington's aligner produces two alternatives. In the final, mixed set, ALINE makes a single mistake in *daughter/thugatēr*, in which it posits a dropped prefix rather than a syncopated syllable; in all other cases, it is right on target. Overall, ALINE clearly performs better than Covington's aligner.

Somers (1999) tests one version of his algorithm, CAT, on the same set of cognates. CAT employs binary, rather than multivalued, features. Another important characteristic is that it pre-aligns the stressed segments in both sequences. Since CAT distinguishes between individual consonants, in some cases it produces more accurate alignments than Covington's aligner. However, because of its pre-alignment strategy, it is guaranteed to produce wrong alignments in all cases when the stress has moved in one of the cognates. For example, in the Spanish/French pair *cabeza/cap*, it aligns [p] with [θ] rather than [b] and fails to align the two [a]'s. The problem is even more acute for closely related languages that have different stress rules.[8] In contrast, ALINE does not even consider stress, which, in the context of diachronic phonology, is too volatile to depend on. Except for the single case of *daughter/thugatēr*, ALINE produces better alignments than Somers's algorithm.

6 Future Directions

The goal of my current research is to combine the new alignment algorithm with a cognate identification procedure. The alignment of cognates is possi-

[8]For example, stress regularly falls on the initial syllable in Czech and on the penultimate syllable in Polish, while in Russian it can fall anywhere in the word.

ble only after the pairs of words that are suspected of being cognate have been identified. Identification of cognates is, however, an even more difficult task than the alignment itself. Moreover, it is hardly feasible without some kind of pre-alignment between candidate lexemes. A high alignment score of two words should indicate whether they are related. An integrated cognate identification algorithm would take as input unordered wordlists from two or more related languages, and produce a list of aligned cognate pairs as output. Such an algorithm would be a step towards developing a fully automated language reconstruction system.

Acknowledgments

I would like to thank Graeme Hirst, Elan Dresher, Steven Bird, and Carole Bracco for their comments. This research was supported by Natural Sciences and Engineering Research Council of Canada.

References

John H. Connolly. 1997. Quantifying target-realization differences. *Clinical Linguistics & Phonetics*, 11:267–298.

Michael A. Covington. 1996. An algorithm to align words for historical comparison. *Computational Linguistics*, 22(4):481–496.

Michael A. Covington. 1998. Alignment of multiple languages for historical comparison. In *Proceedings of COLING-ACL'98: 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pages 275–280.

Bennett L. Fox. 1973. Calculating the *K*th shortest paths. *INFOR – Canadian Journal of Operational Research and Information Processing*, 11(1):66–70.

Daniel Gildea and Daniel Jurafsky. 1996. Learning bias and phonological-rule induction. *Computational Linguistics*, 22(4):497–530.

Osamu Gotoh. 1982. An improved algorithm for matching biological sequences. *Journal of Molecular Biology*, 162:705–708.

Steven Lee Hartman. 1981. A universal alphabet for experiments in comparative phonology. *Computers and the Humanities*, 15:75–82.

Grzegorz Kondrak. 1999. Alignment of phonetic sequences. Technical Report CSRG-402, University of Toronto. Available from ftp.cs.toronto.edu/csri-technical-reports.

Joseph B. Kruskal. 1983. An overview of sequence comparison. In David Sankoff and Joseph B. Kruskal, editors, *Time warps, string edits, and macromolecules: the theory and practice of sequence comparison*, pages 1–44. Reading, Mass.: Addison-Wesley.

Peter Ladefoged. 1995. *A Course in Phonetics*. New York: Harcourt Brace Jovanovich.

Roy Lowrance and Robert A. Wagner. 1975. An extension of the string-to-string correction problem. *Journal of the Association for Computing Machinery*, 22:177–183.

Eugene W. Myers. 1995. Seeing conserved signals. In Eric S. Lander and Michael S. Waterman, editors, *Calculating the Secrets of Life*, pages 56–89. Washington, D.C.: National Academy Press.

John Nerbonne and Wilbert Heeringa. 1997. Measuring dialect distance phonetically. In *Proceedings of the Third Meeting of the ACL Special Interest Group in Computational Phonology (SIGPHON-97)*. Available at http://www.cogsci.ed.ac.uk/sigphon/.

B. John Oommen. 1995. String alignment with substitution, insertion, deletion, squashing, and expansion operations. *Information Sciences*, 83:89–107.

T. F. Smith and Michael S. Waterman. 1981. Identification of common molecular sequences. *Journal of Molecular Biology*, 147:195–197.

Harold L. Somers. 1998. Similarity metrics for aligning children's articulation data. In *Proceedings of COLING-ACL'98: 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pages 1227–1231.

Harold L. Somers. 1999. Aligning phonetic segments for children's articulation assessment. *Computational Linguistics*, 25(2):267–275.

Robert A. Wagner and Michael J. Fischer. 1974. The string-to-string correction problem. *Journal of the Association for Computing Machinery*, 21(1):168–173.